Truck Platooning: week 10

Changyao Zhou, Jiaxin Pan

March 3, 2022

1 Introduction of the project

The goal of this project is to chase a chosen target vehicle by predicting the steering angle and throttle input of the ego-vehicle, given the velocity of both vehicles and images taken from cameras.

The control values (steering angle and throttle) computed by Model Predictive Control (MPC) are used as ground-truth. MPC is an advanced method of process control that is used to control a process while satisfying a set of constraints[WIK]. It takes a prediction horizon instead of a single time step into account, and aims to get an optimal control result minimizing the objective function within the prediction horizon. For learning the MPC controller, we take [MPC] as reference.

The project is designed to be realized in the following steps:

1). Control the ego-vehicle with Model Predictive Control (MPC) in the Carla simulator

2). Given x, y-location, yaw angle and velocity of both vehicles, train a fully connected neural network to control ego-vehicle by using the MPC output as ground-truth.

3). Given object velocity, ego-velocity and RGB image, train a neural network to control the ego-vehicle by using the MPC output as ground-truth

4). Further improvement by changing the input of the neural network

2 Overview of this week

Last week we used the image rendering strategy to collect the off-trajectory dataset. By cropping the images with rectangular shape and a larger field of view, we have generally eliminated the black holes on the image edges. Then we used the rendered image dataset to train our CNN and got a relatively stable result.

This week we first tried to use the stereo system to estimate the depth map. Then with the estimated depth map and RGB images form the camera, we could do image rendering and collect the dataset like last week. Then we trained the CNN for predicting the relative transformation between two vehicles.

Until now, we have tried models with different architectures. Then we used two metrics, route completion and infraction count, to evaluate and compare the performance of different models.

All results of models mentioned above were tested with the 'online' mode in CARLA, where the target vehicle is set in 'autopilot' mode. All the videos are stored under https://syncandshare.lrz.de/getlink/fiF186hzjLCFn9PcVQ7Tr8Qp/week10.

3 Stereo System

The brain uses this binocular disparity to extract depth information from the two-dimensional retinal images which are known as stereopsis. Similarly, here we used stereo system to estimate the depth map [OPE]. The general principle is shown in figure1. Two cameras are placed close to each other with a small distance. The stereo cameras perceive the depth map by using stereo disparity. Disparity refers to the difference in image location of an object seen in two different cameras. Then with disparity map, we can get estimated depth Z with

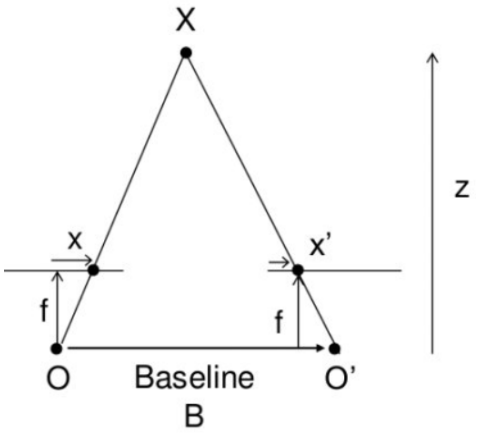

Figure 1: Stereo System Principle

$$disparity = x - x' = \frac{Bf}{Z} \tag{1}$$

where x and x are the distance between points in image plane corresponding to the scene point 3D and their camera center. B is the baseline distance between two cameras (which we know) and f is the focal length of camera (already known)

In our case, we place the two RGB-cameras as stereo cameras with a close distance 0.5m and the y-position of two cameras are y = 0, y = 0.5. Then we can use opencv functions ***cv.StereoBM_create*** to compute the disparity map and estimate depth map with the formular above. From figure 2 - 5, we can see two orginal RGB images from two stereo cameras, disparity map, and the estimated depth map.

Figure 2: original middle image

Figure 3: original right image

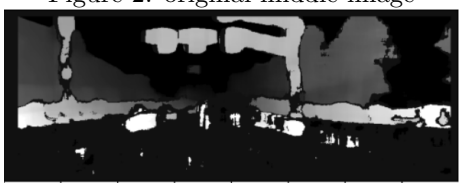

Figure 4: disparity map

Figure 5: estimated depth map

We can see form figure5, the estimate depth map is not so precise, especially in the sky part, which has extremely large depth. The sky part is white, which means estimated depth is close to $+\infty$. According to the formula above, this will lead to black holes in the back-projection image. To fix this problem, we set the estimated depth map of the sky part as a fixed limited value, so that the back-projection image can look reasonable and consistent.

Now we get the estimated depth map through stereo system, the following process is same as last week. We collected rectangular images (800x300) with a fov of 130° first for on-trajectory routes. Then we sampled form the offset range and did the image rendering. At last, we cropped the rectangle image from the top and get a smaller image with size 250x400 to fix the black hole problem. Then the cropped image is resized to 128x128, which is the same size as the CNN input.The rendered images are shown in figure 6 - 8. We can see the rendered images without offset and small angle rotation

seem consistent. But there are still black holes with lateral offset. Unfortunately, when we tried to apply longitudinal offset, even small offset would lead to large black holes in the render images, so we decided not to collect such samples.

Figure 6: rendered image without offset

Figure 7: rendered image with rotation 10°

Figure 8: rendered image with lateral offset 2m

4 Data Collection and Training

4.1 Stereo-Estimation Case

When collecting off-trajectory samples, we set the range of rotation $(-10°, 10°)$ and the range of lateral offset (-2m, 2m). Then the images are fed to the CNN model as inputs, together with the velocities of the two vehicles, the outputs are Δx, Δy, Δyaw between the target- and ego vehicles.

Similar to last week, we first collected 8,000 on-trajectory images from 8 routes in Town 03 and Town 04, and then generated 32000 images with lateral offsets and rotational offset. After training for 40000 samples, the validation loss for Δx, Δy, Δyaw was 2.9e-2, 8.7e-2 and 1.25e-3 respectively, which are much larger than the loss we got last week with real-depth-map inputs.

However, as we can see, the quality of the rendered images, which use the estimated depth map based on stereo system, is not so stable and might contain much noise. So the prediction of control values are not so accurate as before, and the car following performance is not so stable as well. We have tested the performance of rained model on other trajectories in town 1,3,4,5. Later we will evaluate the model performance with other metrics. The videos are saved in https://syncandshare.lrz.de/getlink/fiLSuWhj1mxcSn7tGjnbnAgU/stereo

4.2 On-Trajectory Case

To compare the performance of different models, we tried to train a basic model only with a basic dataset, which only contains 16000 on-trajectory samples from 8 trajectories. After training the model, the validation loss can reach 2e-2 in total. The validation loss for Δx, Δy, Δyaw was 2.7e-2, 2.9e-2 and 1.6e-3 respectively. However, since the training dataset only contains on-trajectory samples, Once the ego-vehicle deviates from the track of the target vehicle, it's hard to go back to the correct trajectory and it will get lost. Later we will evaluate the model performance with random impulses.

5 Evaluation Metrics

Until now, we have obtained different models through different approaches. But how can we evaluate the car following performance? Especially when the ego-vehicle can target vehicle for the whole trajectory, it's hard for us to qualitatively evaluate the performance. To evaluate performance of car following quantitatively, we introduced two evaluation metrics, route completion and infraction count.

5.1 Route Completion

Route Completion(RC) means the ratio between the trajectory that the ego-vehicle can follow and the original trajectory of target vehicle. Here we compute the ratio in frames. We use two conditions to determine whether the car stopped following the preceding car:

1). The distance between two vehicles is larger than 20m.

2). The velocity of ego-vehicle is less than 0.3m/s while the velocity of target vehicle is larger that 1.0m/s.

If any of the above situations occur during driving, it is determined that the car has stopped following. Then we can simply compute the route completion with

$$RC = \frac{currentframe}{totalframe} \tag{2}$$

5.2 Infraction Count

Infraction Count(IC) represent whether the car collides with the environment or other objects during the following process. Here we use the lane invasion detector and the collision detector in CARLA simulator [CAR], ***sensor.other.lane_invasion and sensor.other.collision***. Every time the collision or invasion is detected, we compute the IC with

$$IC = 0.8^n * 0.5^m \tag{3}$$

where n is the number of lane invasions, m is the number of object collisions. In this way, we can know the larger the result, the less the number of conflicts.

5.3 Performance Evaluation

With the two metrics mentioned above, we ran the inference and measured the performance for the trained models before. We have measured the performance for the following models:

1). CNN-MLP model trained with basic on-trajectory dataset

2). CNN-MLP model trained with stereo-based dataset

3). CNN-MLP model trained with image-rendering-based dataset

We used 10 new trajectories in town 1,3,4,5 to evaluate the two metrics, the result can be seen in table1.

Table 1: Evaluation of Car Following Performance

Number	Town	StartingPoint	CNN-MLP(basic)		CNN-MLP(stereo-based)		CNN-MLP(render-based)	
			IC	RC	IC	RC	IC	RC
1	01	0	4.4e-2	32.2%	5.0e-1	14.8%	2.0e-1	14.1%
2	01	100	2.1e-1	29.2%	8.3e-2	42.3%	3.4e-2	92.5%
3	01	150	1.00	37.0%	2.7e-2	33.3%	3.3e-1	46.8%
4	03	0	6.4e-1	37.4%	2.0e-1	17.7%	4.1e-1	87.6%
5	03	112	5.4e-2	47.4%	2.0e-1	14.5%	4.0e-1	16.1%
6	04	5	5.0e-1	61.4%	5.0e-1	62.6%	2.1e-1	100%
7	04	368	1.9e-3	25.0%	2.2e-2	14.0%	8.5e-2	100%
8	05	2	1.00	14.2%	4.1e-1	14.1%	3.2e-1	12.2%
9	05	100	3.4e-2	46.7%	2.7e-2	42.8%	4.8e-3	100%
10	05	150	8.2e-2	44.6%	6.4e-1	18.3%	6.8e-5	100%

6 Conclusion and future work

This week we have tried to use stereo-system to estimate depth map and do image rendering based on estimated depth map, so that we won't need depth camera in reality. Then we trained the CNN as before. In addition, we also trained a CNN with only the on-trajectory samples, which can be used as a base version to compare with others.

So far we have tried different approaches to collect dataset and different architectures of models. Then we introduced two metrics to evaluate car following performance. There are still two more models we want to evaluate, the CNN-MLP model trained with direct CARLA dataset and the end-to-end model.

In the next week, we will continue to evaluate, compare different models and draw conclusions. Also, we would prepare for the presentation and report submission.

References

[CAR] Carla: Collision and Lane Invasion sensors. https://carla.readthedocs.io/en/latest/ref_sensors/#lane-invasion-detector.

[MPC] Model Predictive Control. https://github.com/WuStangDan/mpc-course-assignments.

[OPE] OpenCV: Depth Map from Stereo Images. https://docs.opencv.org/3.4/dd/d53/tutorial_py_depthmap.html.

[WIK] Model Predictive Control. https://en.wikipedia.org/wiki/Model_predictive_control.